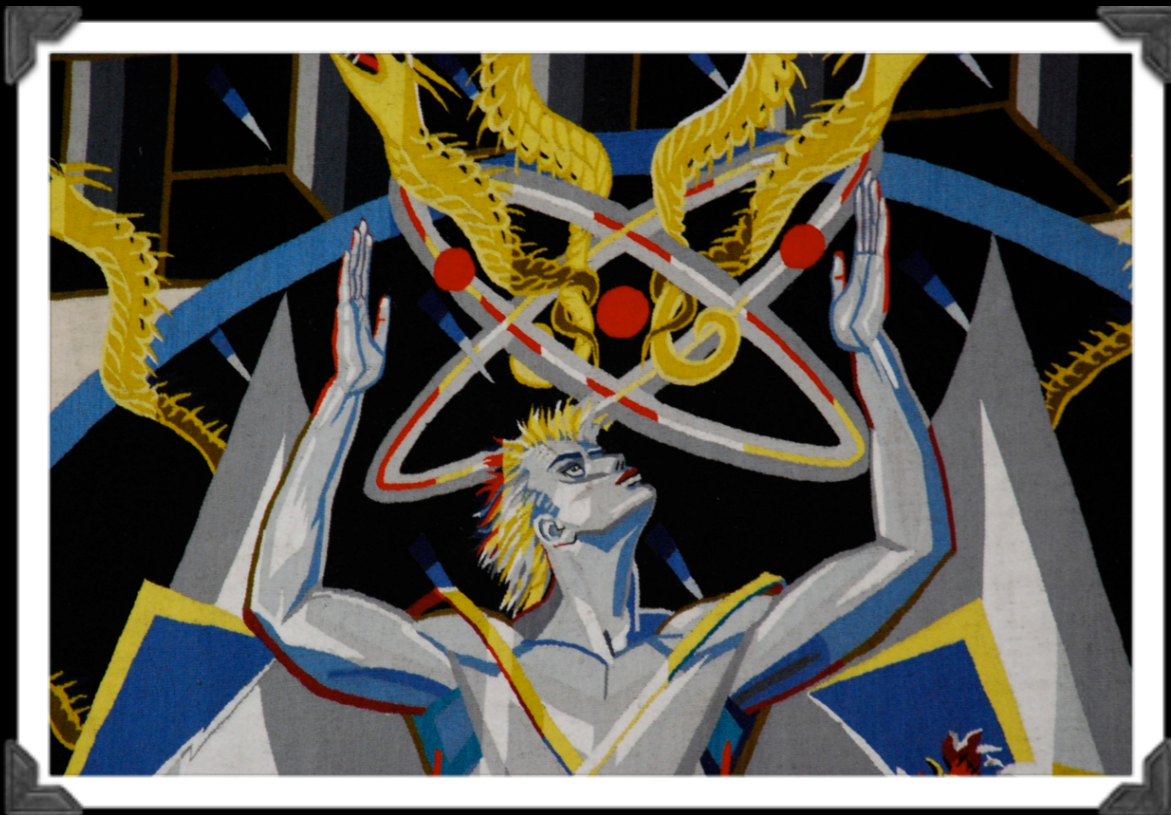


# Scaling Websites with Amazon's EC2

Rusty Conover ([rconover@infogears.com](mailto:rconover@infogears.com))

<http://www.infogears.com>



# About Me

- Co-founder of InfoGears
- NYC via Montana and NJ.
- Computer Science
- Price comparison engine
- @rusty\_conover



# Audience Survey

# Audience Survey

Amazon

# Audience Survey

Amazon

Linux

# Audience Survey

Amazon

Linux

HTTP

# Audience Survey

Amazon

Linux

HTTP

DNS

# Audience Survey

Amazon

Linux

HTTP

DNS

Apache

# Audience Survey

Amazon

Linux

HTTP

DNS

Apache

TCP/IP

# Audience Survey

Amazon

Linux

HTTP

DNS

Apache

TCP/IP

Megabits

# Audience Survey

Amazon

“Expires”

Linux

Header

HTTP

DNS

Apache

TCP/IP

Megabits

# Audience Survey

Amazon

Linux

HTTP

DNS

Apache

TCP/IP

Megabits

“Expires”

Header

Regular  
Expressions

# Audience Survey

Amazon

Linux

HTTP

DNS

Apache

TCP/IP

Megabits

“Expires”

Header

Regular  
Expressions

Proxy Servers

# Audience Survey

Amazon

Linux

HTTP

DNS

Apache

TCP/IP

Megabits

“Expires”

Header

Regular  
Expressions

Proxy Servers

Hash

Algorithms

# Audience Survey

Amazon

Linux

HTTP

DNS

Apache

TCP/IP

Megabits

“Expires”

Header

Regular  
Expressions

Proxy Servers

Hash

Algorithms

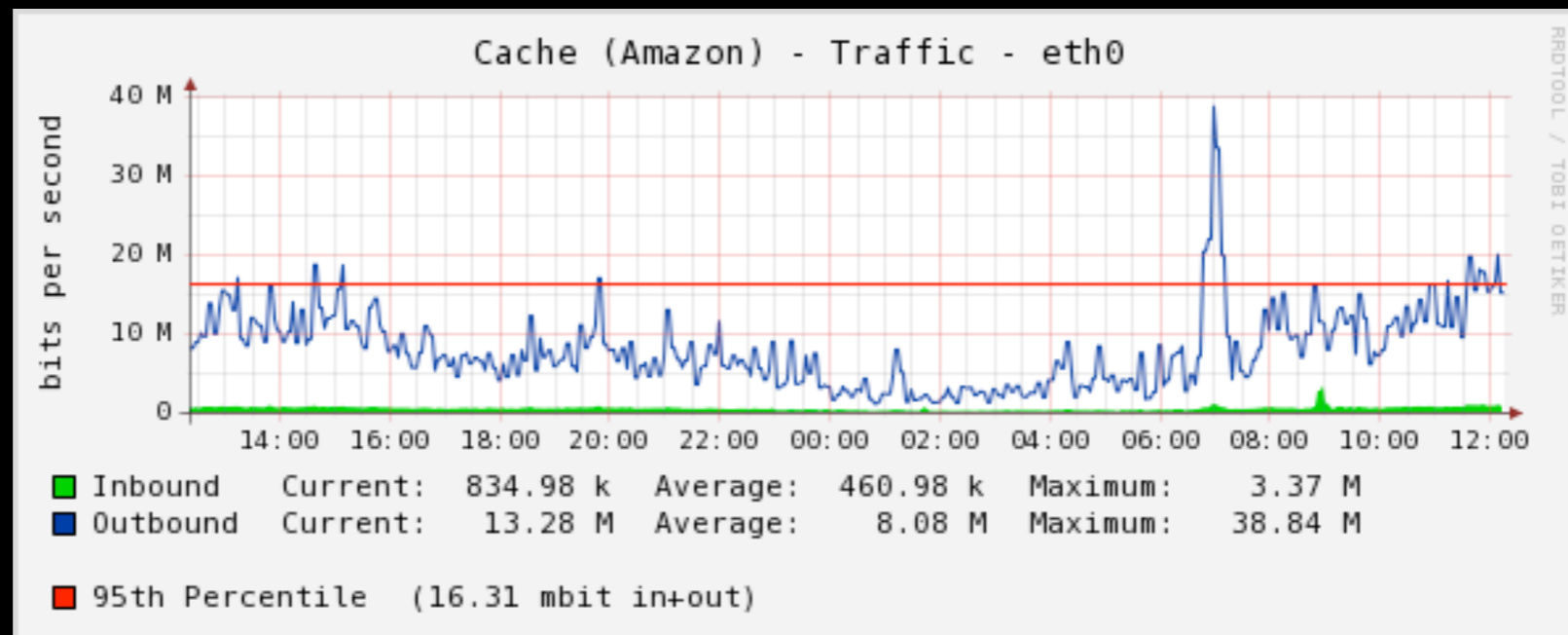
TCP Windowing

# The Problems

- If the site's slow, users leave dissatisfied which often means lost sales.
- Bandwidth is relatively expensive.
- It's hard to anticipate needs.
- Resources (time, money and people) are always limited.

# Other Concerns

- Thousands of people all want to watch your video at once.
- Viral campaigns
- The unexpected media mention.



# The solutions



Content Distribution Networks



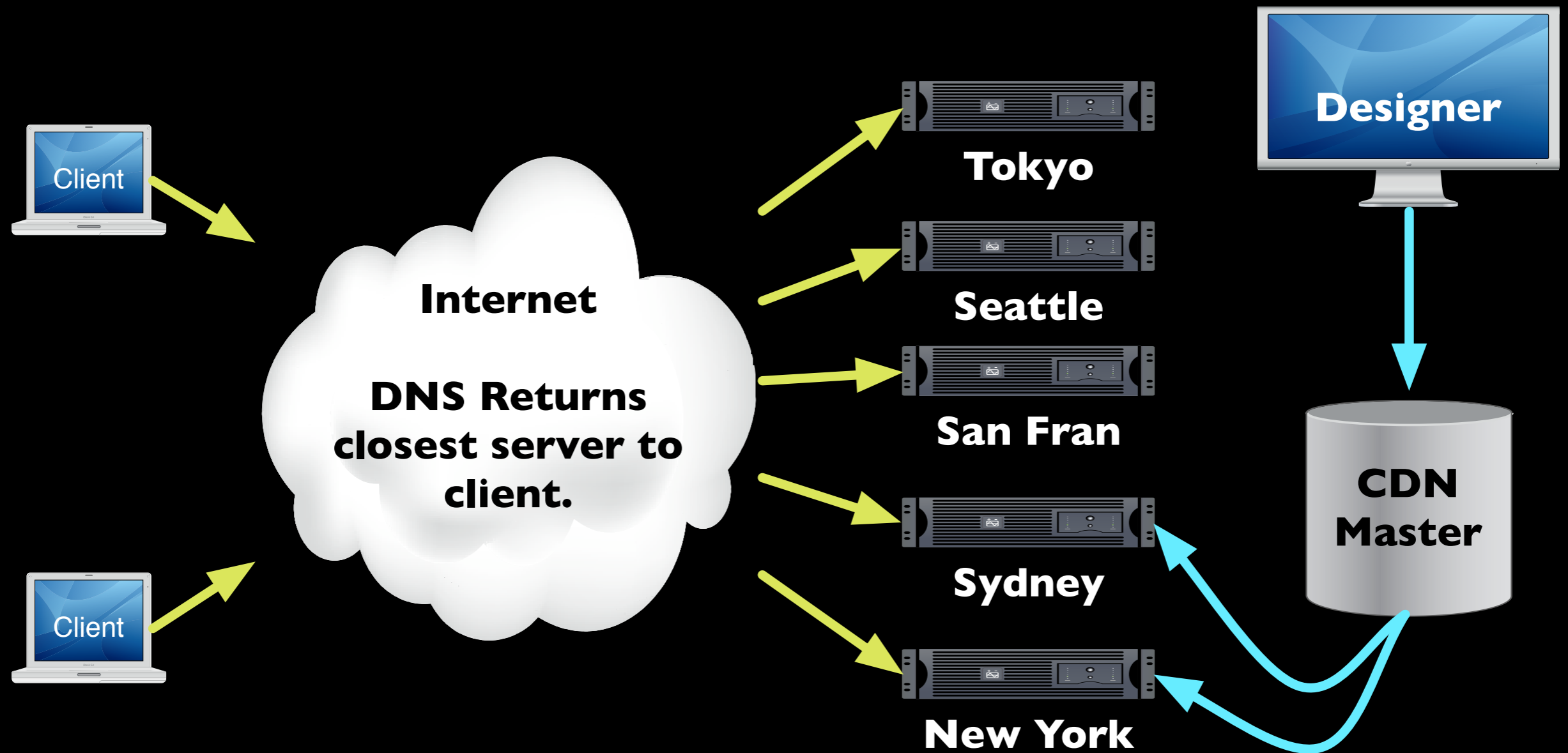
Load Balancers



Reverse Proxying



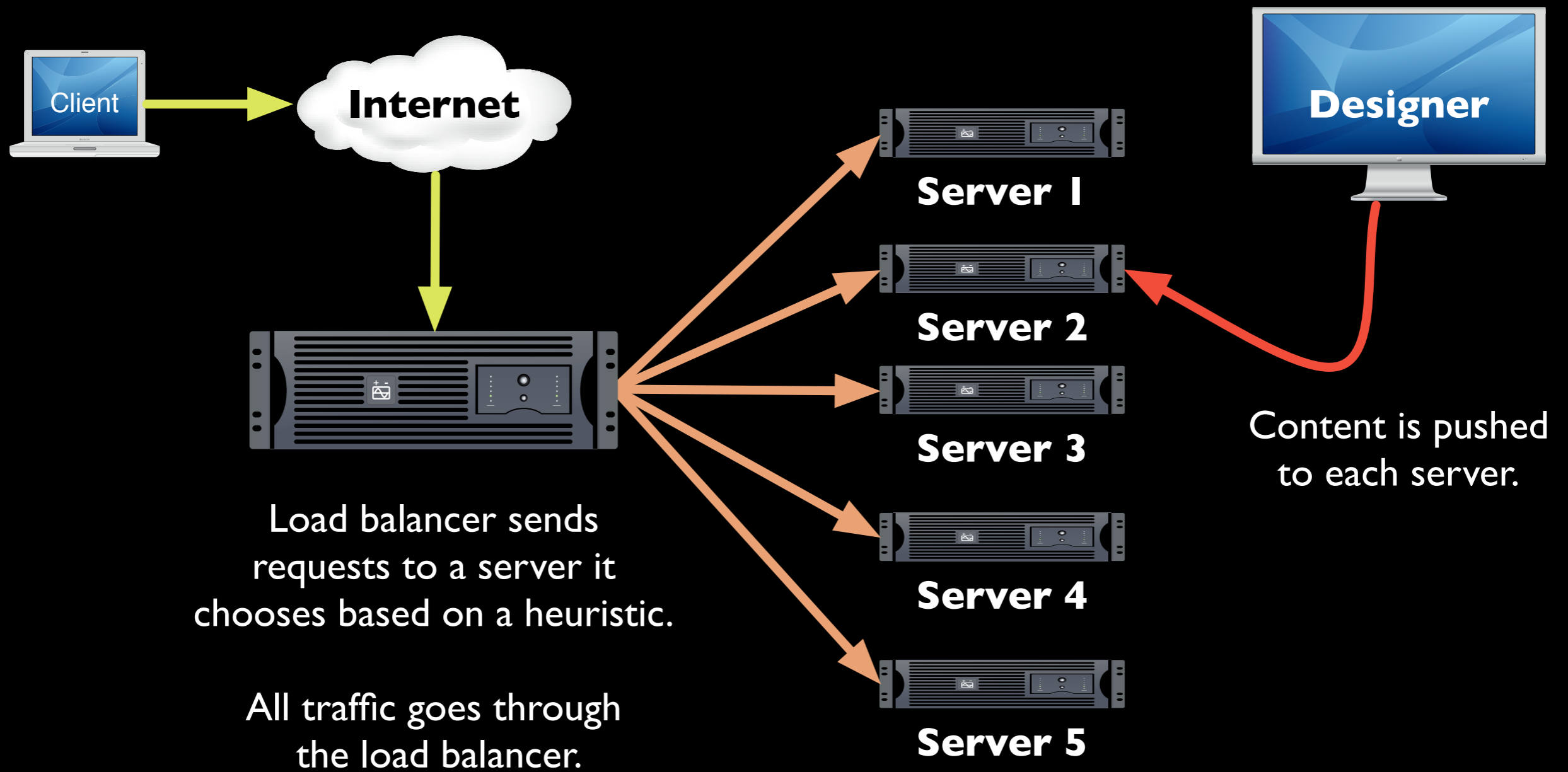
# Content Distribution



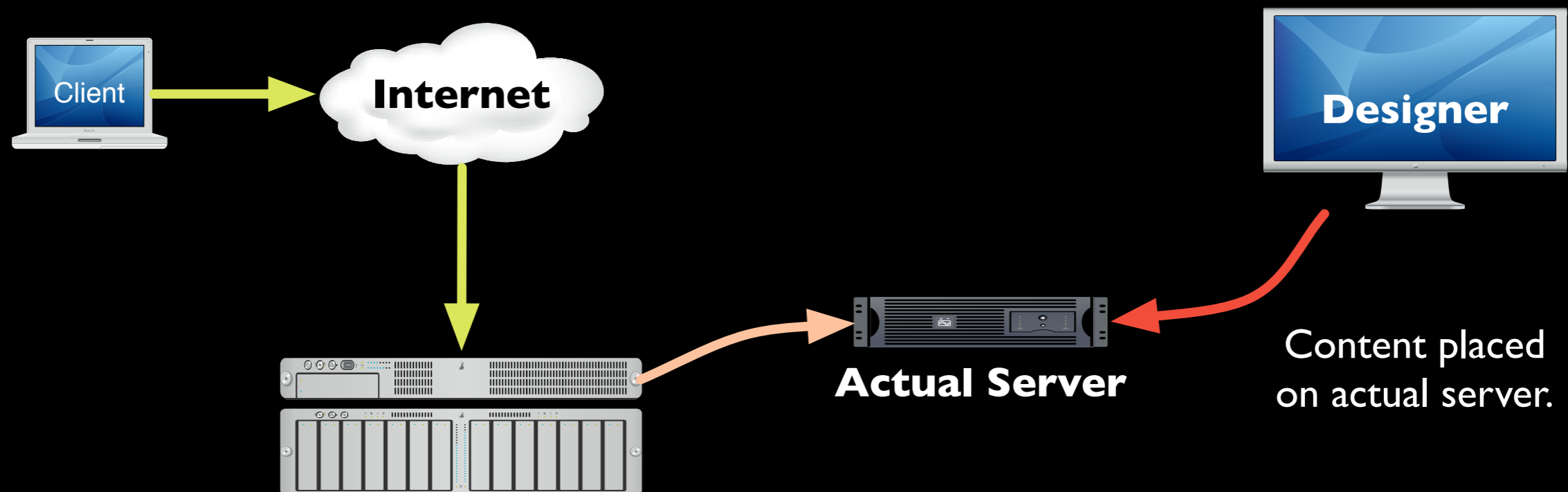
Example Providers:



# Load Balancer

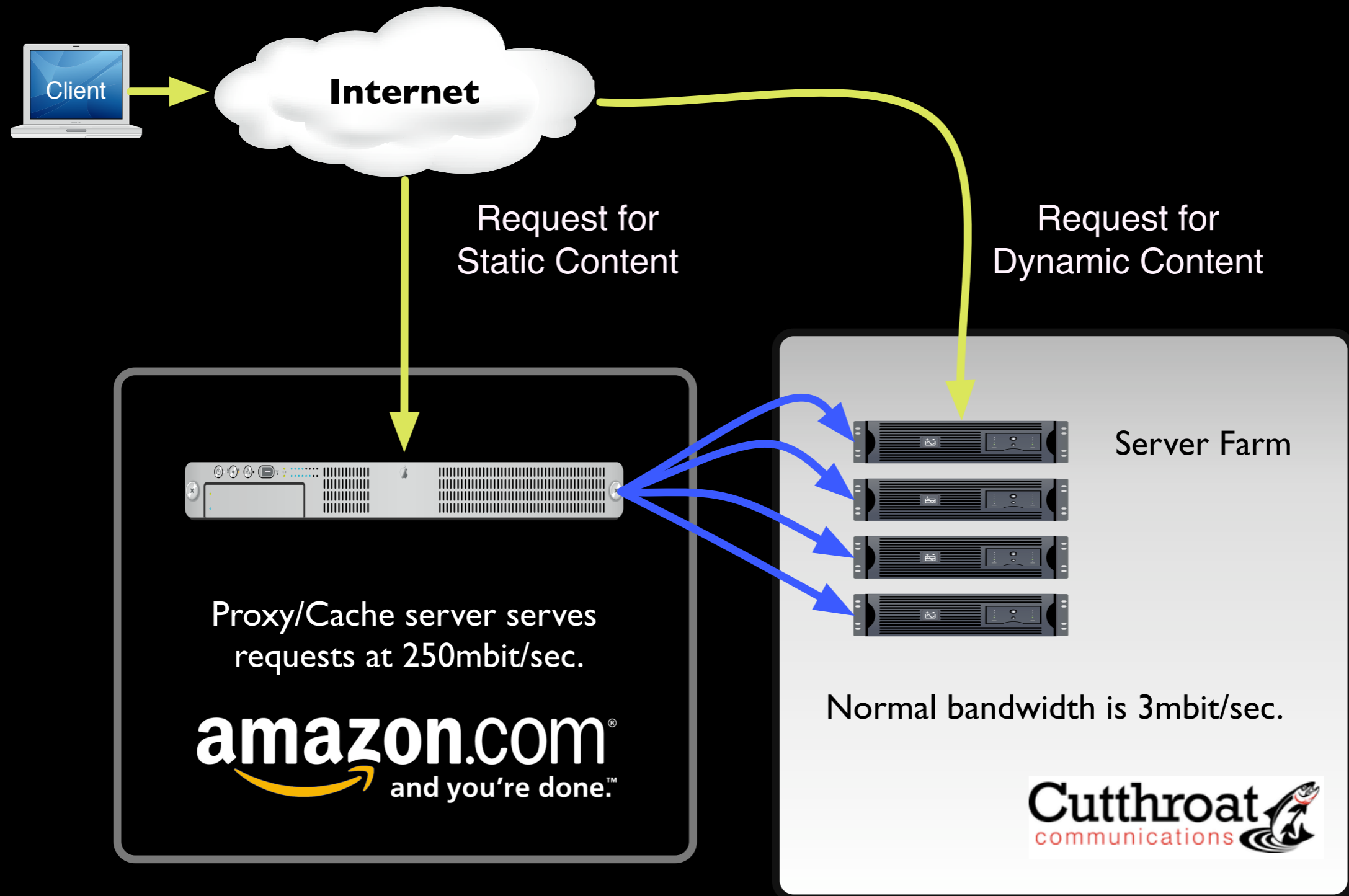


# Reverse Proxy



Proxy server checks if request is cached, if not it pulls from the actual server and caches for future requests.

# Our Solution



# Amazon's EC2

Allows you to have as many virtual machines as you'd like.

- Basic - 1.0 Ghz 2007 Xeon 1.7 GB RAM, 160 GB storage.
- Large - 8 CPUs, 15 GB Ram, 1680 GB storage, 64-bit platform.

# Bandwidth...

- Amazon EC2's bandwidth is about 250 - 1000Mbps.
- You're only billed for what you use. Making it cheap, but different than what you're used to.
- Amazon is located closer to peering points. There are East Coast and Europe DC's.

# Pricing

It's all usage based, with discounts.

Bandwidth	CPU Usage
\$0.10 per GB (incoming)	\$0.03 per hour Small
\$0.17 per GB (first 10 TB outgoing)	\$0.12 per hour Large
Free between S3 and EC2	Billed the entire time the machine is online.

# Limitations of EC2

- When an instance is shutdown all of the disks are wiped. Cache is cleared.
- There are no guarantees that a particular machine will remain up.



# So what's it good for?

1. Parallel processing tasks without building a server farm.
2. Building cache servers to serve your content on quickly and cheaper than you can.
3. Bragging about how your infrastructure is in “the cloud”.

# How to scale

Most websites have both static and dynamic content

Serving separately will increase response time.

Static	Dynamic
Images, Videos, CSS, JS	ASP, PHP, Perl, CGI, HTML
Files that don't change.	Files that do change
Larger	Smaller

# Dynamic Proxy/Cache

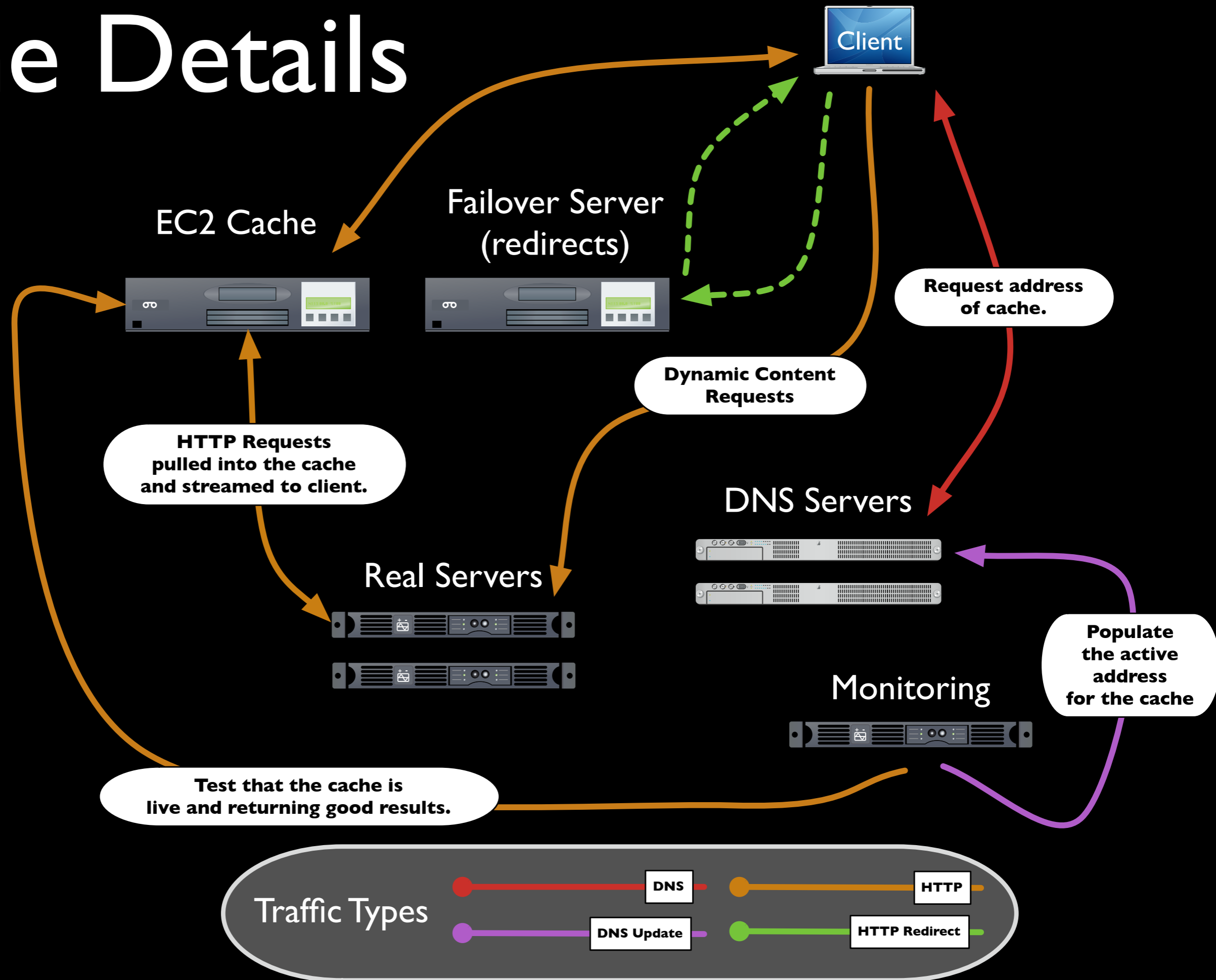
- Static requests are only sent to the reverse proxy/cache.
- Redirects to the real server if there an error

<http://www.foo.com/movie.mpg>

to

<http://cache.foo.com/movie.mpg>

# The Details



# A Little About EC2

- Amazon provides a number of disk images, like ISOs for base installs.
- Fedora Core & Windows.
- You can customize your own install but start with something small.

# Amazon Images

```
$ ec2-describe-images -o self -o amazon
```

```
IMAGE ami-3c03e655 cache7/image.manifest.xml 314456711494  
available private
```

```
IMAGE ami-20b65349 ec2-public-images/fedora-core4-  
base.manifest.xml amazon available public
```

# Create an instance

Create an instance:

```
$ ec2-run-instances ami-2103e648
```

Showing current instances:

```
$ ec2-describe-instances
```

```
RESERVATION r-9a8076f3 314456711494 default
INSTANCE    i-4603fc2f  ami-3c03e655
             ec2-72-44-35-86.z-2.compute-1.amazonaws.com
             domU-12-31-35-00-09-C2.z-2.compute-1.internal
             running      0      m1.small
             2008-02-20T22:07:13+0000
```

Stopping and cleaning up an instance:

```
$ ec2-terminate-instances i-4603fc2f
```

```
INSTANCE    i-4603fc2f  terminated  terminated
```

# DNS

EC2 **will go down**, when you least expect it.

You don't want the users to get errors and you don't want to be sending requests to a down server for very long.

Use dynamic DNS updates and keep very short TTL times for the records. Or EC2's static addresses.

Monitoring and DNS code needs to be reliable, use more than once separate network.

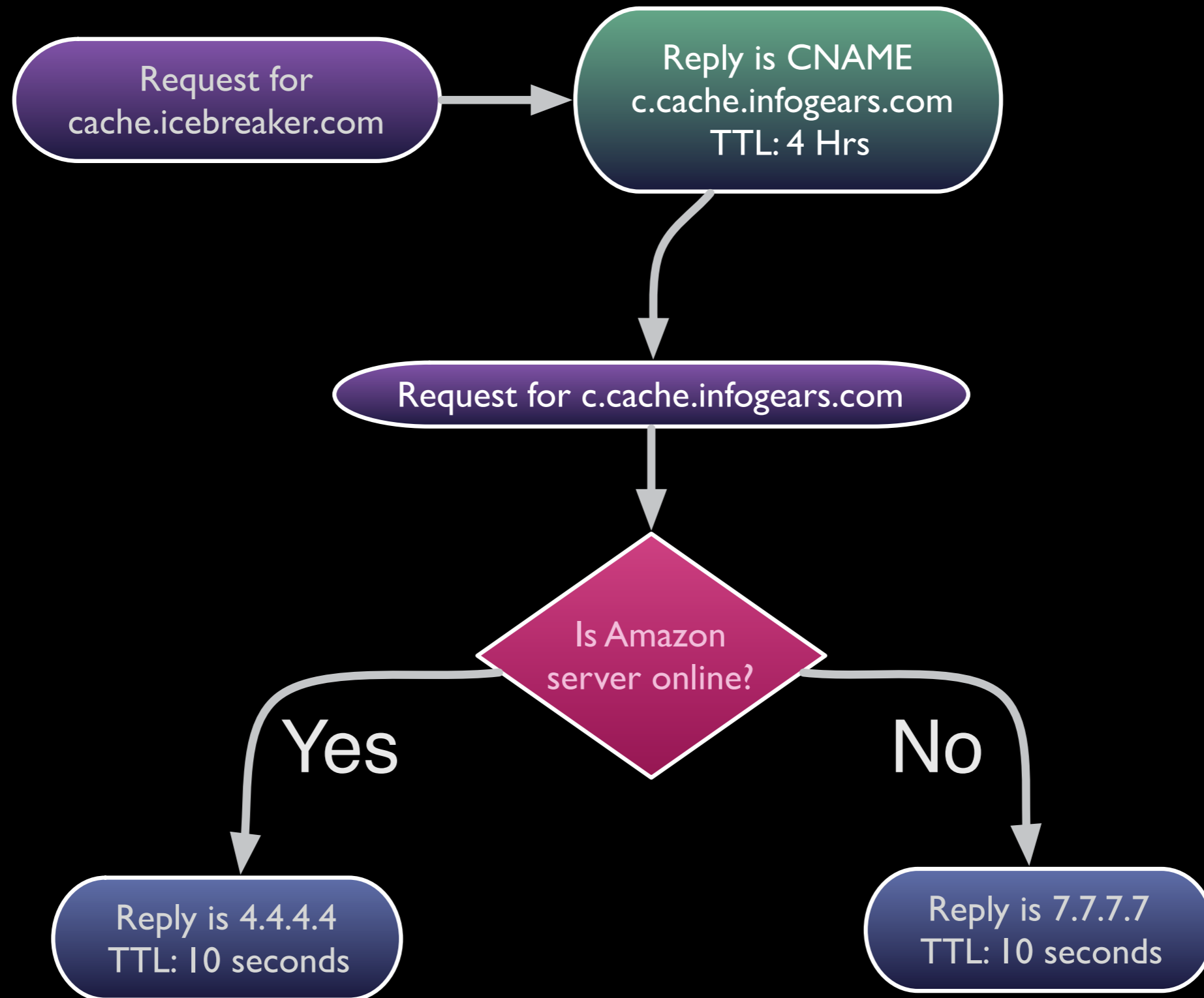
# DNS Redirection

If you host more than one website, generally you don't want to setup instances for every domain.

Setup one caching instance, and then create CNAME records for all of your other domains.

For instance to cache requests at [www.prolitegear.com](http://www.prolitegear.com) I can use [cache.prolitegear.com](http://cache.prolitegear.com) which is a CNAME for [c.cache.infogears.com](http://c.cache.infogears.com).

# DNS Flowchart



# The Cache Stack

apache

bind

snmpd

sshd

ntpd

Fedora Core 4 w/Updates



# Setup

- You need to build in **mod\_cache**, **mod\_proxy** & **mod\_rewrite**.
- Keep the server as small as possible, no PHP or mod\_perl.
- You can set it up to use a memory or disk cache.

```
./configure --enable-cache --enable-mem-cache --enable-disk-cache --enable-proxy --enable-proxy-http --enable-status --enable-info --enable-rewrite --disable-proxy-ftp --disable-proxy-ajp --disable-proxy-balancer --enable-deflate --disable-cgi --disable-cgid --disable-userdir --disable-alias --disable-cgid --disable-actions --disable-negotiation --disable-asis --disable-info --disable-filter --disable-static --enable-headers
```



# Logging

It's good to judge cache hits to make sure your cache is working.

```
LogFormat "%{Host}i %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %{Age}\" proxy
```

The Age header contains the age of the cached result in seconds, if not found it logs “-”.

Logs should be sent back to reliable storage every so often.



# Proxy



`ProxyRequests Off`

```
<Proxy *>  
    Order deny,allow  
    Allow from all  
</Proxy>
```

- Make sure you don't make an open proxy.
- Our proxy requests will only be the result of rewrite rules.



# Rewrite



```
RewriteMap lowercase int:tolower  
RewriteMap cachehost txt:/usr/local/apache2/conf/cache-host.map  
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$  
RewriteCond ${cachehost:%1} ^(.+)$  
RewriteRule ^/(.*\.(gif|jpg|jpeg|png))$ http://%1/$1 [P,L,NC]  
RewriteRule ^/$ http://www.infogears.com [R,L]
```

The rewrite rule is what changes the cached url into the real url to pull for the request.

The map file just lists the cache host name [TAB] destination host name.



# Host Mapping

#	Destination Host	Source Host
	<code>static-cache.gearbuyer.com</code>	<code>static.gearbuyer.com</code>
	<code>images-cache.gearbuyer.com</code>	<code>images.gearbuyer.com</code>
	<code>cache.gearbuyer.com</code>	<code>www.gearbuyer.com</code>



# Cache



```
CacheEnable disk /  
CacheRoot /mnt/cache  
CacheDirLevels 3  
CacheDirLength 1  
CacheIgnoreCacheControl Off  
CacheDefaultExpire 7200  
CacheMaxExpire 604800  
CacheMaxFileSize 2000000000
```

Make sure that the cache root exists before Apache starts, otherwise it won't start. /mnt is a good place.

Make sure you have the correct permissions so Apache can write to the directory.

Change the directory levels and limits to suit your needs.



# Scaling

```
ServerLimit 600
StartServers      20
MinSpareServers  20
MaxSpareServers  60
MaxClients       500
MaxRequestsPerChild  0

MaxKeepAliveRequests 1000
KeepAlive On
KeepAliveTimeout 10
SendBufferSize 98303
```



Since you're serving static requests it won't take much RAM to scale out more processes.

Keep alive connections should persist as they prevent another TCP handshake.



# Monitoring

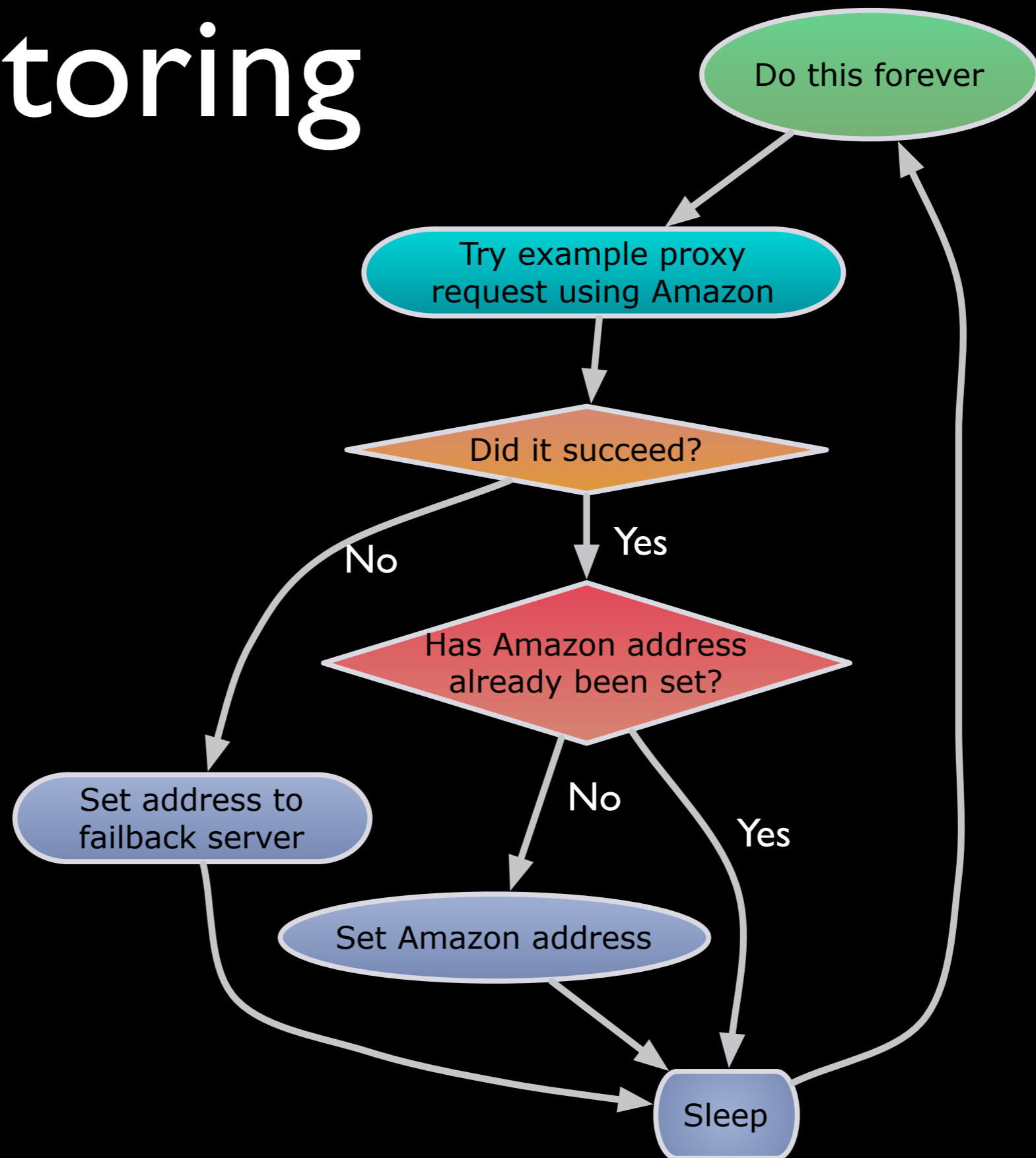
Monitoring is important to make sure that EC2 can reach your servers, and your EC2 server is still running.

I use Perl for this since it has everything I need: a way to update DNS and a way to send web requests.

SNMP Traffic Monitoring is also essential.



# Monitoring





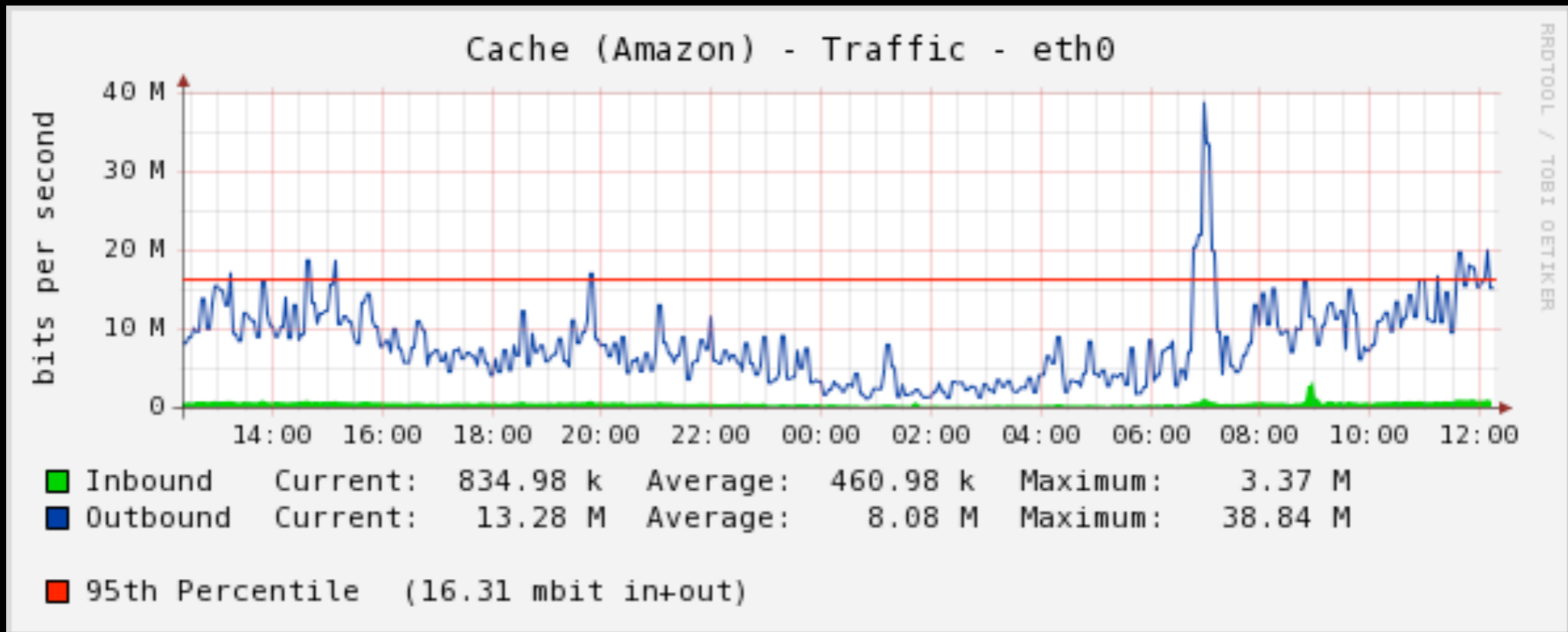
# Hit Rate

Set Expires header on everything you can.

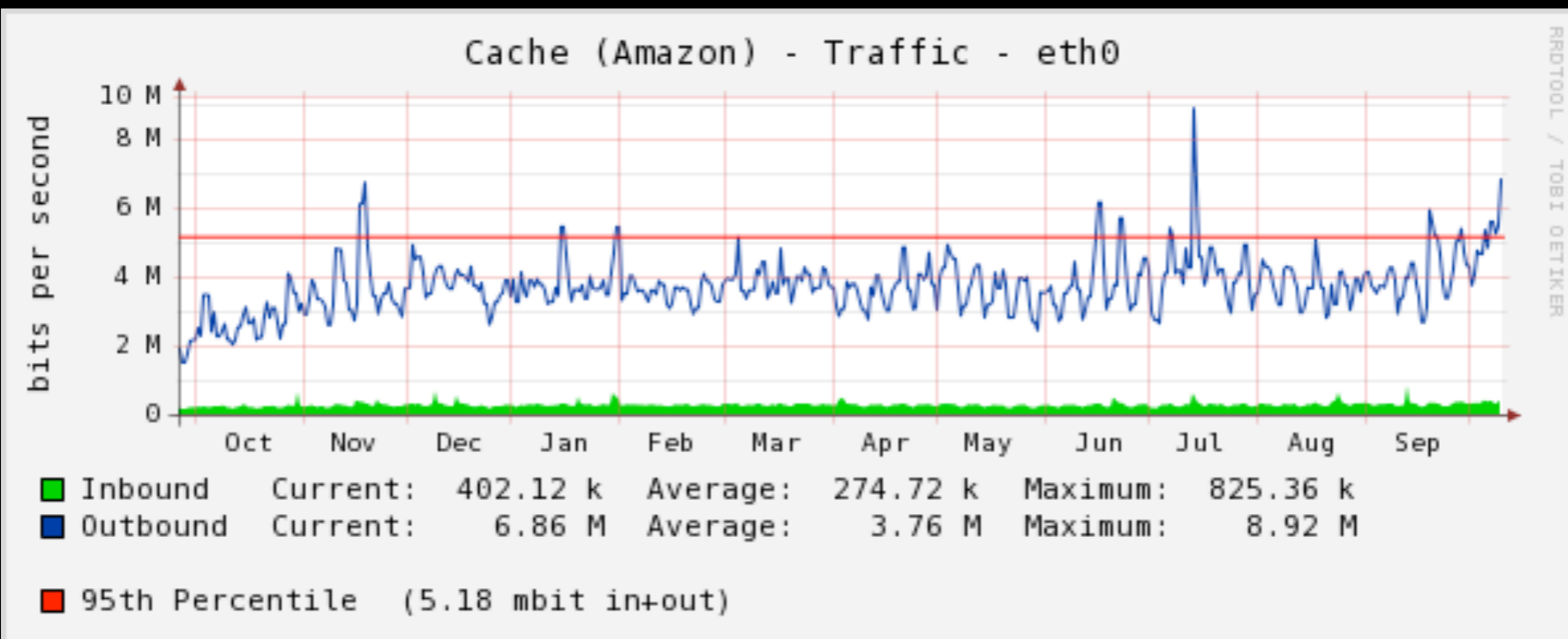
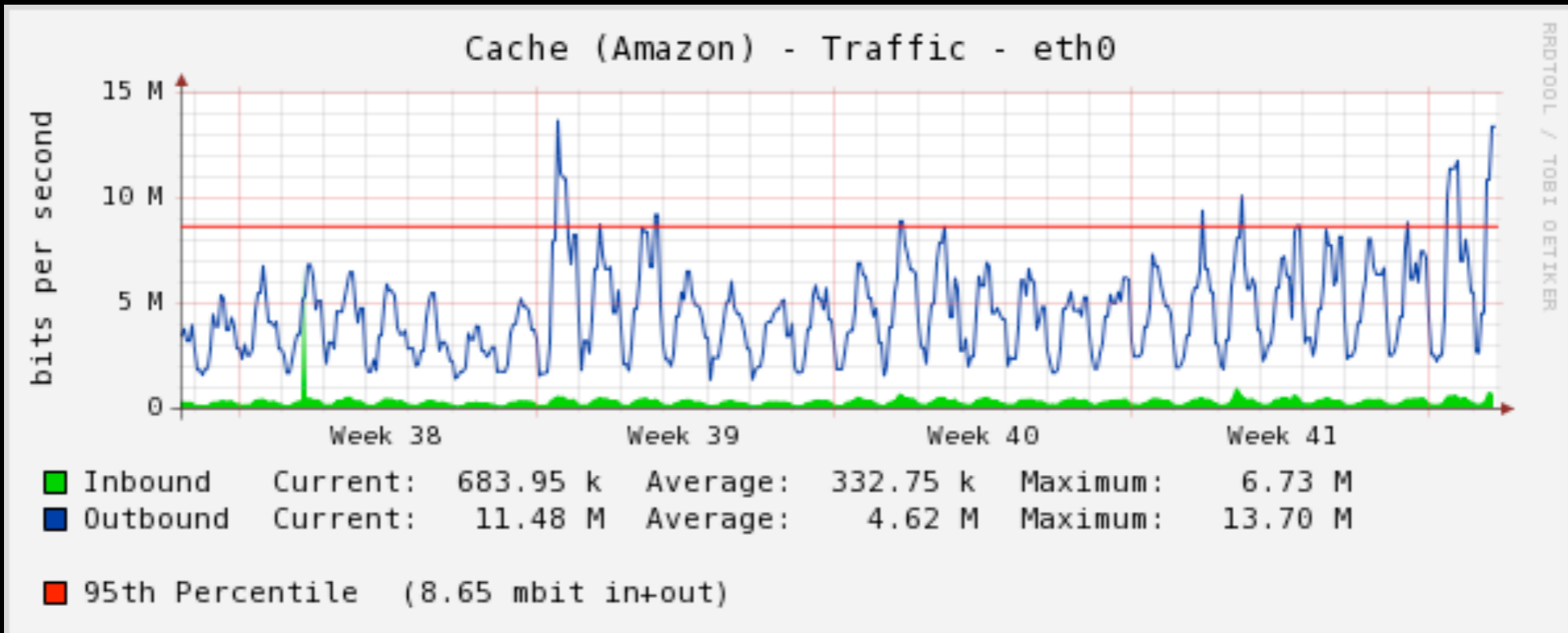
```
ExpiresActive on
ExpiresByType image/gif "access plus 8 hours"
ExpiresByType image/png "access plus 8 hours"
ExpiresByType image/jpeg "access plus 8 hours"
ExpiresByType text/css "access plus 8 hours"
ExpiresByType application/x-javascript "access plus 8 hours"
ExpiresByType application/x-shockwave-flash "access plus 8 hours"
ExpiresByType video/x-flv "access plus 8 hours"
ExpiresByType application/pdf "access plus 8 hours"
```

You can force refreshes by doing a reload,  
or using `wget --no-cache`

# SNMP Monitoring



These graphs are generated by Cacti



Date	Hits	Misses	GB	Mean Hit Size	Hit Ratio	Hits Per Second
Oct 1, 2008	44,156,082	4,559,317	936	22	89.7%	16.8
Nov 1, 2008	61,403,833	3,475,474	1,291	22	94.3%	23.3
Dec 1, 2008	66,850,143	3,896,923	1,342	21	94.2%	25.4
Jan 1, 2009	67,415,819	3,877,521	1,302	20	94.2%	25.6
Feb 1, 2009	59,252,656	3,817,855	1,107	20	93.6%	22.5
Mar 1, 2009	67,494,160	4,391,207	1,452	23	93.5%	25.7
Apr 1, 2009	58,472,655	4,823,061	1,277	23	91.8%	22.2
May 1, 2009	57,608,518	4,719,774	1,327	24	91.8%	21.9
Jun 1, 2009	55,105,350	4,658,961	1,326	25	91.5%	21.0
Jul 1, 2009	53,533,558	4,993,568	1,626	32	90.7%	20.4
Aug 1, 2009	59,870,784	5,642,222	1,371	24	90.6%	22.8
Sep 1, 2009	62,931,209	5,519,812	1,490	25	91.2%	23.9

# References

- Page Load Time:

- [http://www.die.net/musings/page\\_load\\_time/](http://www.die.net/musings/page_load_time/)

- Amazon EC2 Reference:

- <http://docs.amazonwebservices.com/AWSEC2/2007-08-29/GettingStartedGuide/>

- Amazon Web Services:

- <http://aws.amazon.com>

- Perl

- <http://www.perl.org>

- Bind Dynamic Updates

- <http://www.isc.org/sw/bind/arm93/Bv9ARM.ch04.html>

- Apache

- <http://httpd.apache.org>



# Amazon Cloudfront

- Price: \$0.01 per 10k req, \$0.17/gb traffic + S3 costs
- Sept: \$62/hits, \$253.30/traffic = \$316 not counting S3 costs.
- Have to preload all resources to S3. Cache has about 2.2 million objects in 36 gigs.